Conditional Logic

/*Conditional Logic focuses on how to write statements that can behave differently depending on the data encountered during statement execution. The mechanism used for conditional logic in SQL statements is the case expression, which can be utilized in select, insert, update, and delete statements.

Conditional logic is simply the ability to take one of several paths during program execution.*/

```
SELECT
    first_name,
    last_name,
    CASE
        WHEN active = 1 THEN 'ACTIVE'
        ELSE 'INACTIVE'
    END activity_type
FROM
    customer;
```

```
SELECT
    first_name,
    last_name,
    CASE
        WHEN active = 1 THEN 'This customer is ACTIVE'
        ELSE 'This customer is INACTIVE'
    END customer_status
FROM
    customer;
```

```
SELECT
    first_name,
    last_name,
    CASE
        WHEN active = 1 THEN 'This customer is ACTIVE'
        ELSE 'This customer is INACTIVE'
    END customer_status
FROM
    customer
WHERE
    active = 'This customer is INACTIVE';
```

/*Searched case Expressions

The case expression demonstrated earlier is an example of a searched case expression, which has the following syntax:

```
CASE
      WHEN C1 THEN E1
      WHEN C2 THEN E2
      ...
      WHEN CN THEN EN
      [ELSE ED]
END
```

When the case expression is evaluated, the when clauses are evaluated in order from top to bottom; as soon as one of the conditions in a when clause evaluates to true, the corresponding expression is returned, and any remaining when clauses are ignored. If none of the when clause conditions evaluates to true, then the expression in the else clause is returned.*/

```
SELECT
    c.first_name,
    c.last_name,
    CASE
        WHEN active = 0 THEN 0
        ELSE (SELECT
                COUNT(*)
            FROM
                rental AS r
            WHERE
                r.customer_id = c.customer_id)
    END num_of_rentals
FROM
    customer AS c;

SELECT
    c.first_name,
    c.last_name,
    CASE
        WHEN active = 0 THEN 'This customer is INACTIVE'
        ELSE (SELECT
                COUNT(*)
            FROM
                rental AS r
            WHERE
                r.customer_id = c.customer_id)
    END num_of_rentals
FROM
    customer AS c;
```

```
SELECT
    c.first_name,
    c.last_name,
    CASE
        WHEN active = 1 THEN (SELECT
                COUNT(*)
            FROM
                rental AS r
            WHERE
                r.customer_id = c.customer_id)
        ELSE 'This customer is INACTIVE'
    END num_of_rentals
FROM
    customer AS c;
```

/*Simple case Expressions

The simple case expression is quite similar to the searched case expression but is a bit less flexible.

```
CASE V0
     WHEN V1 THEN E1
     WHEN V2 THEN E2
     ...
     WHEN VN THEN EN
     [ELSE ED]
END
```

Simple case expressions are less flexible than searched case expressions because you can’t specify your own conditions, whereas searched case expressions may include range conditions, inequality conditions, and multipart conditions using and/or/not*/

/*Examples of case Expressions

Result Set Transformations*/

```
SELECT
    MONTHNAME(rental_date) AS rental_month,
    COUNT(*) AS num_of_rentals
FROM
    rental
WHERE
    rental_date BETWEEN '2005-05-01' AND '2005-08-01'
GROUP BY MONTHNAME(rental_date);

SELECT
    MONTHNAME(rental_date) AS rental_month,
    COUNT(*) AS num_of_rentals
FROM
    rental
WHERE
    rental_date BETWEEN '2005-05-01' AND '2005-08-01'
GROUP BY 1;
```

```
SELECT
    SUM(CASE
        WHEN MONTHNAME(rental_date) = 'May' THEN 1
        ELSE 0
    END) AS may_rentals,
    SUM(CASE
        WHEN MONTHNAME(rental_date) = 'June' THEN 1
        ELSE 0
    END) AS june_rentals,
    SUM(CASE
        WHEN MONTHNAME(rental_date) = 'July' THEN 1
        ELSE 0
    END) AS july_rentals
FROM
    rental
WHERE
    rental_date BETWEEN '2005-05-01' AND '2005-08-01';
```

/*Checking for Existence

Here’s a query that uses multiple case expressions to generate three output columns, one to show whether the actor has appeared in G-rated films, another for PG-rated films, and a third for NC-17-rated films*/

```
SELECT
    a.first_name,
    a.last_name,
    CASE
        WHEN
            EXISTS( SELECT
                    1
                FROM
                    film_actor AS fa
                        INNER JOIN
                    film AS f ON fa.film_id = f.film_id
                WHERE
                    fa.actor_id = a.actor_id
                        AND f.rating = 'G')
        THEN
            'Yes'
        ELSE 'No'
    END AS g_actor,
    CASE
        WHEN
            EXISTS( SELECT
                    1
                FROM
                    film_actor as fa
                        INNER JOIN
                    film as f ON fa.film_id = f.film_id
```

```
                WHERE
                    fa.actor_id = a.actor_id
                        AND f.rating = 'PG')
        THEN
            'Yes'
        ELSE 'No'
    END pg_actor,
    CASE
        WHEN
            EXISTS( SELECT
                    1
                FROM
                    film_actor as fa
                        INNER JOIN
                    film as f ON fa.film_id = f.film_id
                WHERE
                    fa.actor_id = a.actor_id
                        AND f.rating = 'NC-17')
        THEN
            'Yes'
        ELSE 'No'
    END nc17_actor
FROM
    actor as a
WHERE
    a.last_name LIKE 'S%'
        OR a.first_name LIKE 'S%';
```

```
/*Query uses a simple case expression to count the number of copies
in inventory for each film and then returns either 'Out Of Stock',
'Scarce', 'Available', or 'Common'*/


SELECT
    f.title,
    CASE (SELECT
            COUNT(*)
        FROM
            inventory i
        WHERE
            i.film_id = f.film_id)
        WHEN 0 THEN 'Out Of Stock'
        WHEN 1 THEN 'Scarce'
        WHEN 2 THEN 'Scarce'
        WHEN 3 THEN 'Available'
        WHEN 4 THEN 'Available'
        ELSE 'Common'
    END film_availability
FROM
    film AS f;
```

/***Division-by-Zero Errors**

When performing calculations that include division, you should always take care to ensure that the denominators are never equal to zero. MySQL simply sets the result of the calculation to null*/

```
select 100/0;


SELECT
    c.first_name,
    c.last_name,
    SUM(p.amount) as tot_payment_amt,
    COUNT(p.amount) as num_payments,
    SUM(p.amount) / CASE
        WHEN COUNT(p.amount) = 0 THEN 1
        ELSE COUNT(p.amount)
    END as avg_payment
FROM
    customer as c
        LEFT OUTER JOIN
    payment p ON c.customer_id = p.customer_id
GROUP BY c.first_name , c.last_name;
```

/*Conditional Updates

When updating rows in a table, you sometimes need conditional logic to generate a value for a column.*/

```
UPDATE customer
SET
    active = CASE
        WHEN
            90 <= (SELECT
                    DATEDIFF(NOW(), MAX(rental_date))
                FROM
                    rental r
                WHERE
                    r.customer_id = customer.customer_id)
        THEN
            0
        ELSE 1
    END
WHERE
    active = 1;

select * from customer;
```

/*Handling Null Values

While null values are the appropriate thing to store in a table if the value for a column is unknown, it is not always appropriate to retrieve null values for display or to take part in expressions.*/

```
SELECT
    c.first_name,
    c.last_name,
    CASE
        WHEN a.address IS NULL THEN 'Unknown'
        ELSE a.address
    END AS address,
    CASE
        WHEN ct.city IS NULL THEN 'Unknown'
        ELSE ct.city
    END AS city,
    CASE
        WHEN cn.country IS NULL THEN 'Unknown'
        ELSE cn.country
    END country
FROM
    customer AS c
        LEFT OUTER JOIN
    address AS a ON c.address_id = a.address_id
        LEFT OUTER JOIN
    city AS ct ON a.city_id = ct.city_id
        LEFT OUTER JOIN
    country AS cn ON ct.country_id = cn.country_id;
```

```
/*For calculations, null values often cause a null result*/

SELECT (7 * 5) / ((3 + 14) * null);

/*When performing calculations, case expressions are useful for
translating a null value into a number (usually 0 or 1) that will
allow the calculation to yield a non-null value.*/
```

/*Exercise - 1

Rewrite the following query, which uses a simple case expression, so that the same results are achieved using a searched case expression. Try to use as few when clauses as possible.

```
SELECT name,

CASE name

WHEN 'English' THEN 'latin1'

WHEN 'Italian' THEN 'latin1'

WHEN 'French' THEN 'latin1'

WHEN 'German' THEN 'latin1'

WHEN 'Japanese' THEN 'utf8'

WHEN 'Mandarin' THEN 'utf8'

ELSE 'Unknown'

END character_set

FROM language;*/


SELECT
    name,
    CASE
        WHEN name IN ('English' , 'Italian', 'French', 'German') THEN
'latin1'
        WHEN name IN ('Japanese' , 'Mandarin') THEN 'utf8'
        ELSE 'Unknown'
    END AS character_set
FROM
    language;
```

/*Exercise - 2

Rewrite the following query so that the result set contains a single row with five columns

(one for each rating). Name the five columns G, PG, PG_13, R, and NC_17.

```
mysql> SELECT rating, count(*)
-> FROM film
-> GROUP BY rating;
+--------+----------+
| rating | count(*) |
+--------+----------+
| PG     | 194      |
| G      | 178      |
| NC-17  | 210      |
| PG-13  | 223      |
| R      | 195      |
+--------+----------+*/


SELECT
    SUM(CASE
        WHEN rating = 'G' THEN 1
        ELSE 0
    END) AS g,
    SUM(CASE
        WHEN rating = 'PG' THEN 1
        ELSE 0
    END) AS pg,
    SUM(CASE
        WHEN rating = 'PG-13' THEN 1
        ELSE 0
    END) AS pg_13,
```

```
    SUM(CASE
        WHEN rating = 'R' THEN 1
        ELSE 0
    END) AS r,
    SUM(CASE
        WHEN rating = 'NC-17' THEN 1
        ELSE 0
    END) AS nc17
FROM
    film;
```